Prophesy: A Web-based Performance Analysis and Modeling System for Parallel and Distributed Applications

Valerie E. Taylor, Xingfu Wu, and Rick Stevens

***Abstract*-- Efficient execution of applications requires insight into how the system features impact the performance of the application. This insight generally results from significant experimental analysis and possibly the development of performance models. This paper proposes the web-based Prophesy system, a performance analysis and modeling infrastructure. Prophesy uses databases that allow for the recording of performance data, system features and application details. Prophesy also includes three automated modeling techniques that facilitates the analysis process. Hence, Prophesy can assist in gaining the needed performance insight based upon one's experience and that of others.**

I. INTRODUCTION

TODAY, parallel and distributed systems are very complex, requiring tools to gain insights into the performance of applications executed on such environments. This paper presents the web-based Prophesy system, a performance analysis and modeling infrastructure that uses databases to aid in gaining this needed insight based upon one's experience and that of others. The Prophesy database allows for the recording of performance data, system features and application details. Prophesy also includes three automated modeling techniques to facilitate the analysis process. As a result, Prophesy can be used to develop models based upon significant data, identify the most efficient implementation of a given function based upon the given system configuration, explore the various trends implicated by the significant data, and predict the performance on a different system.

The remainder of this paper is organized as follows. Section 2 describes the Prophesy framework and main components. Section 3 presents the Prophesy database, and data collection framework. Section 4 presents the automated model builder, and describes the three automated modeling methods. Section 5 summarizes this paper.

This work was supported in part by the National Science Foundation under NSF NGS grant EIA-9974960, a grant from NASA Ames and two NSF ITR grants --- GriPhyN and Building Human Capital.

Valerie E. Taylor, Department of Computer Science, Texas A&M University, College Station, TX 77843 (Email: taylor@cs.tamu.edu).

Xingfu Wu, Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208 (Email: wuxf@ece.nwu.edu).

Rick Stevens, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439.

II. PROPHESY FRAMEWORK

The Prophesy framework consists of three major components: data collection (left section) and data analysis (right section), and three central databases, as illustrated in Figure 1. The data collection component focuses on the automated instrumentation and application code analysis at the level of basic blocks, procedures, and functions. An application code can be instrumented at the basic-block level such that a significant amount of performance information can be gathered to gain insight into the performance relationship between the application, hardware and system software. The resultant performance data is automatically stored in the performance database. Manual data entry is also supported. The data analysis component produces analytical performance models with coefficients, at the granularity specified by the user. The models are developed based upon performance data from the performance database, model templates from the template database, and system characteristics from the systems database.

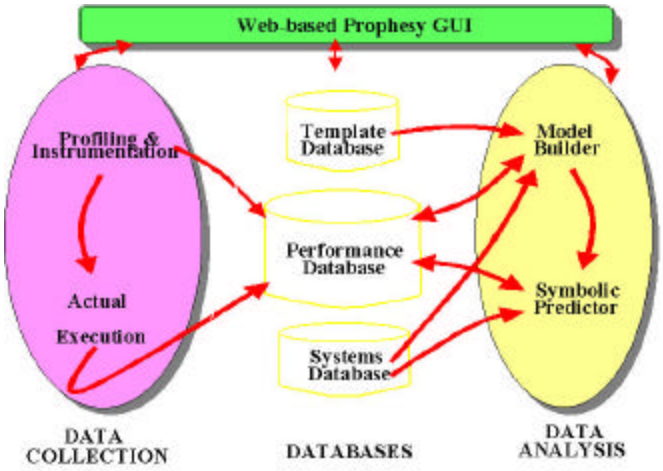

Figure 1 Prophesy framework

An application goes through three stages (instrumentation of the application, performance data collection of many runs, and model development using optimization techniques) to generate an analytical performance model. These models, when combined with data from the system database, can be used by the prediction engine to predict the performance on a different

compute platform. Prophesy is an infrastructure designed to explore the plausibility and credibility of various techniques in performance evaluation (such as scalability, efficiency, speedup, performance coupling between application kernels, etc.) and allow users to use various metrics collectively to bring performance analysis environments to the most advanced level.

III. PROPHESY DATABASE AND DATA COLLECTION

Recall that the Prophesy Database must accommodate queries that lead to the development of performance models, allow for prediction of performance on other systems, and allow for one to obtain insight into methods to improve the performance of the application on a given distributed system. Hence, the database must facilitate the following query types:

- Identify the best implementation of a given function for a given system configuration (identified by the run-time system, operating system, processor organization, etc.). This can be implemented querying the database for comparison of performance data on different systems.
- Use the raw performance data to generate analytical (nonlinear or linear) models of a given function or application; the analytical model can be used to extrapolate the performance under different system scenarios and can be used to assist programmers in optimizing the strategy or algorithms in their programs.
- Use the performance data to analyze application-system trends, such as scalability, speedup, I/O requirements, communication requirements, etc. This can be implemented querying the database to calculate the corresponding formula.
- Use the performance data to analyze user specific metrics such as coupling between functions.

In this section, we present the implementation of the database that accommodates the aforementioned query types and data uses. See [WT01a, WT01b] for detailed information about the Prophesy database and automated instrumentation. Prophesy assumes that applications can be decomposed into modules (or files that comprise an application), which can be further decomposed into functions that can be decomposed into basic units in a hierarchical manner.

A. *Prophesy Database*

The Prophesy database has a hierarchical organization, consistent with the hierarchical structure of the applications. The schema shown in Figure 2 includes all three databases given in Figure 1: performance database, system models database and template database. The entities in the database are organized into four areas: application information, executable information, run information and performance statistics. Descriptions of these four areas are given below.

- **Application Information**: includes entities that give the application name, version number, a short description, owner information and password (such that only the owner can modify or add data for a given application). It is assumed that an application goes through various versions as one adds different functionalities over time. Data are placed into these entities when a new application is being developed.
- **Executable Information**: includes all of the entities related to generating an executable of an application. These entities include details about compilers, libraries and the control flow. The details are given for modules and functions. It is assumed that applications may be developed using multiple languages, such as C, C++, Fortran and HPF. These entities include details about executable, modules, functions, model templates, compilers, libraries and control flow. Data are placed into these entities when a new executable is generated.
- **Run Information:** includes all of the entities related to running an executable, which includes the system information and inputs used for execution. The system may be a single processor, single parallel machine or distributed system. These entities include details about the inputs, system(s) used for execution, system resource and connections between resources. Data are placed into these entities for each run of a given executable.
- **Performance Statistics Information**: includes all of the entities related to the raw performance data collected during execution. These entities include details about the application performance, function performance, basic unit performance and data structure performance. Performance statistics are collected for different levels of granularities. Since a basic unit can be accessed from multiple functions, statistics are collected at the levels of both the function and basic unit. Note that basic unit refers to a code segment that may be of smaller granularity than a function but higher granularity than a basic block.

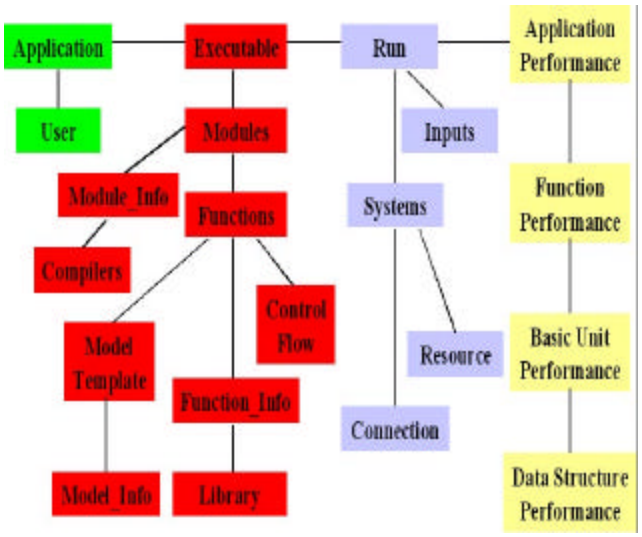

Figure 2. Framework of Prophesy Database Schema

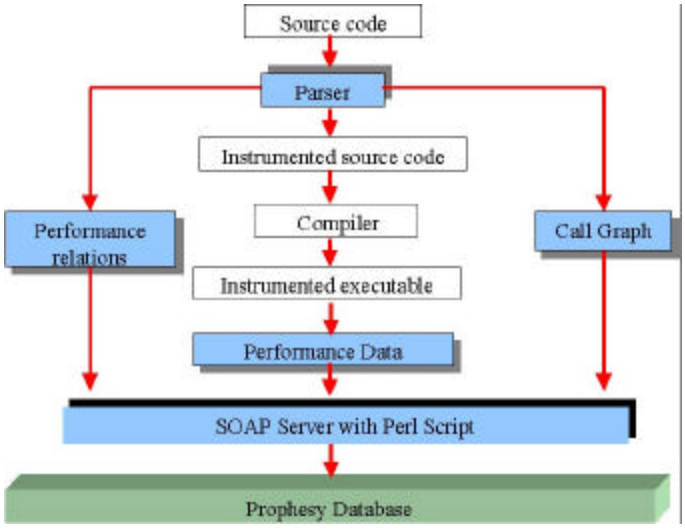

Figure 3. PAIDE framework

B. *Data Collection*

PAIDE (Prophesy Automatic Instrumentation and Data Entry)], shown in Figure 3, is the data collection component of the Prophesy system with the goal of minimizing instrumentation overhead [WT01b]. PAIDE focuses on the automatic instrumentation of codes at the level of basic blocks, procedures, or functions. The default mode consists of instrumenting the entire code at the level of basic loops and procedures. A user can specify that the code be instrumented at a finer granularity than that of loops or identify the particular events to be instrumented. The resultant performance data is automatically placed in the performance database (via ftp) and is used by the data analysis component to produce analytical performance models at the granularity specified by the user. In addition to instrumenting the code, PAIDE generates two files. The first file contains the performance relations that connect each performance event with the corresponding line of code. The second file contains the control flow information of the application.

There are two ways to input the performance data into the database: automatic data entry and interactive data entry. Automatic data entry uses Perl and SOAP to automatically process the performance data files and upload the data into the Prophesy database. The interactive data entry entails using the web interface to manually place the data into the Prophesy database.

IV. AUTOMATED MODEL BUILDER

The main goal of the Prophesy automated model builder is to automatically generate performance models to aid in performance analysis and evaluation of a given application or execution environment. Currently, Prophesy supports the following modeling techniques: curve fitting, parameterization, and composition methods. The first two methods are well-established techniques; the last method, coupling, wad developed by the Prophesy research group [TW01, TW02]. Each model type has its own unique set of options and parameters that a user can easily modify to explore different scenarios, in terms of the applications and execution environment. These three methods are described below.

A. *Curve Fitting Method*

Curve Fitting is a method that uses optimization techniques to develop a model. In this case, the model builder uses a least squares fit when performing the curve fit; this method uses the empirical data found in the Prophesy database to generate the model. The user determines the empirical data and then *Octave* is used to generate the resultant model. The weakness in this model is the lack of exposure of the system terms versus the application terms. The models generated from curve fitting are generally a function of some input parameters of the application and the number of parameters. The system performance, i.e. the operation execution time or communication performance, are clustered together with the coefficients determined by the curve fitting; such parameters are not exposed to the user. The advantage of this method is the fact that only the empirical data is needed to generate the models; no manual analysis is required.

B. *Parameterization Method*

Parameterization is a method that combines manual analysis of the code with system performance measurements. The manual analysis entails hand-counting the number of different operations in the code. It is assumed that this type of analysis is done on kernels or functions that are generally in the range of 100 lines of code or less. With Prophesy, the manual analysis is used to produce an analytical equation with terms that represent the application and the execution environment. Hence, users can explore different application and execution environment scenarios with parameterized models. The only drawback to this method is the manual analysis step that is involved; yet this step is done only once per kernel.

C. *Composition Method*

The composition modeling technique focuses on how to effectively represent the performance of an application in terms of its component kernels or functions. Hence, this technique focuses on the level of functions, for which most applications have only a few major functions. The major goal of the composition method is to use kernel performance models to develop the full application performance models. When developing performance models of applications it is extremely useful to understand the relationships between the different functions that compose the application. Basically, we want to determine how one kernel affects another, whether or not it is a constructive or a destructive relationship. Further, we want to be able to encapsulate this information into a coefficient that can be used in a performance model of the application. As an

example, let's say we have an application composed of three kernels, *kernelA*, *kernelB*, and *kernelC* which operates in a loop. Also, assume that we analyzed each kernel and produced *modelA*, *modelB*, and *modelC* accordingly. The composition method would provide a set of coefficients for the relationships such that the following is satisfied, where T is the execution time for the application:

$$T = ? \, modelA + ? \, modelB + ? \, modelC$$

where the coefficients ?, ?, and ? represents the performance relation between the three kernels that identifies how they should be combined to reflect the performance of the application. See the reference [TW02] for the detailed information about the coupling method.

V. CONCLUSIONS

This paper presented the web-based Prophesy system for performance analysis and modeling of parallel and distributed applications. Prophesy includes automatic instrumentation of applications, a database to hold performance and context information, and an automated model builder for developing performance models. The Prophesy system allows users to gain needed insights into application performance based upon their experience as well as that of others. The Prophesy web interface is given in Figure 4 and can be found at the following URL: http://prophesy.mcs.anl.gov. Currently, we are focused on extending the tool to different application communities.

Figure 4. Prophesy web page

REFERENCES

[TW01] Valerie Taylor, Xingfu Wu, Xin Li, Jonathan Geisler, Zhiling Lan, Mark Hereld, Ivan R. Judson and Rick Stevens, Prophesy: Automating the Modeling Process, *Third Annual International Workshop on Active Middleware Services (invited paper)*, CA, August 2001.

[TW02] Valerie Taylor, Xingfu Wu, Jonathan Geisler, and Rick Stevens, Using Kernel Couplings to Predict Parallel Application Performance, *Proc. of the 11th IEEE International Symposium on High-Performance Distributed Computing (HPDC 2002)*, Edinburgh, Scotland, July 24-26, 2002

[WT01a] Xingfu Wu, Valerie Taylor, J. Geisler, Z. Lan, X. Li, R. Stevens, M. Hereld, I. Judson, Design and Development of Prophesy Performance Database for Distributed Scientific Applications, *Proceedings of the 10th SIAM Conference on Parallel Processing*, March 2001.

[WT01b] Xingfu Wu, Valerie Taylor, and Rick Stevens, Design and Implementation of Prophesy Automatic Instrumentation and Data Entry System, *Proceedings of the 13th IASTED Parallel and Distributed Computing and Systems Conference* (PDCS2001), August 2001.